embodiment there exists an instrumentation program able to ascertain the correct aspects of a software system to control. Also included is a method to allow administrators and end users to view and modify those items to be virtualized by the system.

[0061] In the automated program, the application to be controlled is observed in order to gauge the aspects of control. The automated program is capable of performing this task during the installation process of the application, during run-time of the application, or a combination of both. In the preferred embodiment, the Operating System Guard is embedded in a wrapper application. Post installation, or after one or many uses of the software, the wrapper application will query the Operating System Guard for a detailed list of all of its actions. From this list of actions, the wrapper application will create the configuration files required to load and operate the Operating System Guard on subsequent uses.

[0062] If used as part of the installation process, the Operating System Guard, in the preferred embodiment, will act as a virtual layer allowing the installation to be entered into its environment only. After the installation, all of the files, settings, et. al. can be dumped for reload later. In this way, the installation will leave the original system intact and will have automatically created the necessary configuration files. When used during use of the application, the Operating System Guard is able to record either differential modifications to the environment, or recodify the configuration files.

[0063] The Operating System Guard will pass its information to the wrapper application for post-processing. In the preferred embodiment, in addition to the automatic entries that the system can create, the wrapper application is programmed with operating system specific and application or domain specific knowledge. This knowledge is used to alter the output of the process to reflect known uses of configuration items or other entries. In the preferred embodiment, a rules-based system is employed to compare observed behaviors with known scenarios in order to effect changes to the coding.

[0064] The wrapper application is also used as a viewer and/or editor for the configuration output of the process. This editor, in the preferred embodiment, enables a system administrator to add, edit, or delete items or groups of items from the configuration. In observing the configuration through the editor, the administrator can also make replicas of the configuration, changing specific items as needed to effect application level or user custom changes.

[0065] Referring now to **FIG. 1**, an embodiment of the present invention is illustrated functionally. In this embodiment, two sets of application/user data **60** are illustrated. The Operating System Guard **100** keeps the two instances of the application **50** from interfering with one another. In addition, as explained above, the operating system guard **100** serves as an abstraction layer and as such collects commands and communications between the application software **50** and the actual operating system **10** of the client computer. As illustrated graphically by the arrows, certain commands are between the Operating System Guard and the software application, this is in distinction to typical installations where these commands would instead be acted upon by the operating system itself, resulting in changes to the client computer that might not necessarily be what the operator intended. On the other hand, other commands pass through the Operating System Guard and are then transferred to the Operating System itself.

[0066] While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention encompassed by the appended claims.

[0067] What is claimed is:

1. A method for protecting computer programs from inconsistent run-time conditions, comprising:

running a first application program on a software platform;

providing the first application program with a first run-time environment and a first set of services configured to the first application program;

running a second application program on the same software platform as the first application program;

providing the second application program with a second run-time environment and a second set of services configured to the second application program;

wherein the first run-time environment and first set of services are different from the second run-time environment and second set of services, respectively, thereby providing the first and second application programs with different operating contexts within the same software platform.

* * * * *